Info 206: Computing

Lecture 4

Hash Tables

September 15, 2014

Hash Functions

1. A function $\longrightarrow$ **H(x) = y**
   a. H(x) - the "hash" function
   b. Should be easy to compute
   c. *Compression* - generally maps to a value of smaller size of fixed length (in bits)
2. Ideally, for a function H(x) = y, you **cannot find *x* given *y***
3. Ideally, for any *x*, ***H(x)* is unique** i.e. H(x) != H(x')

(*Ideally* in this case means that it is not "*computationally feasible*")

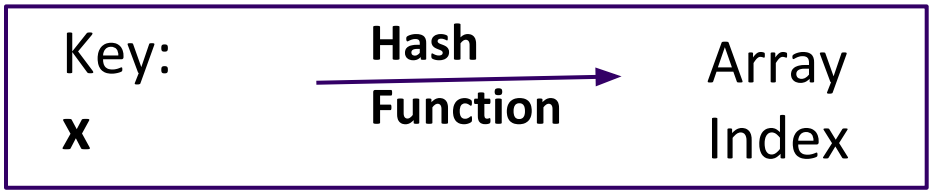
Applications of Hash Functions

- Security - Ex: Saving passwords
- Cryptographic Hash Functions (We'll cover it in more depth later in the semester)
- HASH TABLES

Hash Tables: Basic Idea

- A data structure that stores elements in such a way that it is easy to search them later.
- It can be searched in O(1) time

- It stores values in an array. It uses a hash function to convert the value into an array index.



Index	0	1	2	…	n
Key					
Value					

Hash Tables: Example

- The hash functions used can vary from simple to very complex.
- Example of a very **simple hash function using remainders**:
  - It takes an item and divides it by the table size, returning the remainder as its hash value (***h*(*item*)=*item*%TableSize**).

Ex: Placing 54, 26, 93, 17, 77, and 31 in table of size 11

H(54) = 54%11 = 10

H(26) = 4; H(93) = 5; H(17) = 6; H(77) = 0; H(31) = 9

Index	0	1	2	3	4	5	6	7	8	9	10
Key	77	None	None	None	26	93	17	None	None	31	54

Hash Tables: Generalization

- Hash function can be defined to take any kind of input (large number, strings, file paths, etc) and be mapped to an index that can be used to store key-value pairs.

	Key	Value
0	kim chi	spicy cabbage
1	kreplach	tasty stuffed dough
2	kiwi	New Zealand fruit

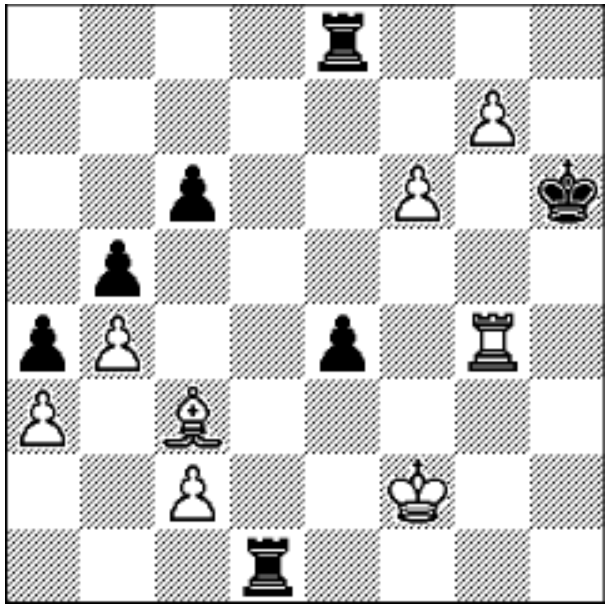
	Key	Value
0	folder1	folder1 Hash Table
1	folder2	folder2 Hash Table
2	file1	file contents

Hash Tables: Facts

- The values are not sorted within the array
- The values are not filled sequentially
- The array is not completely filled

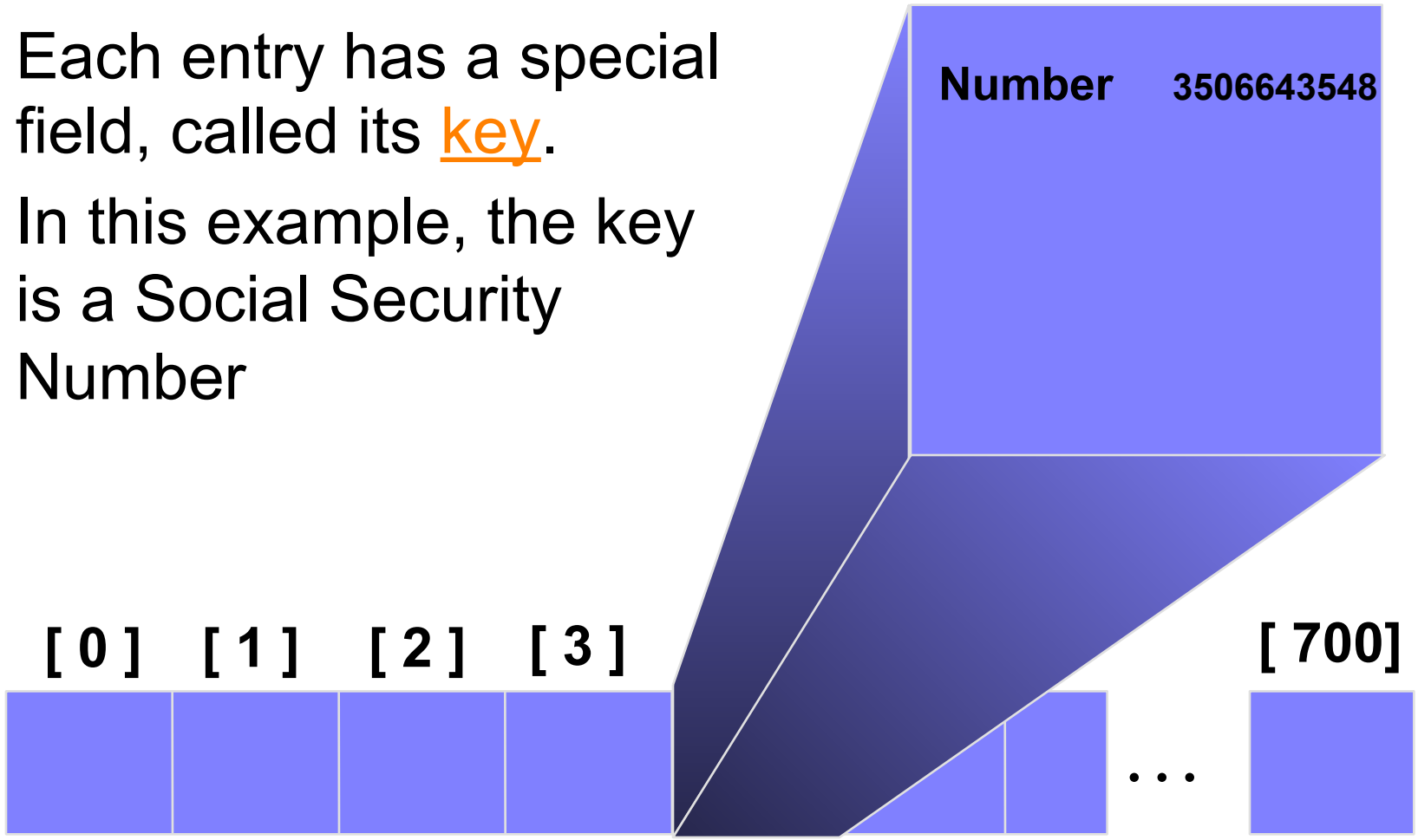
Applications

- When log(*n*) is just too big…
  - Symbol tables in interpreters
  - Real-time databases (in core or on disk)
    - air traffic control
    - packet routing
- When fast lookup is needed
  - Chess endgames
  - Cache Memory



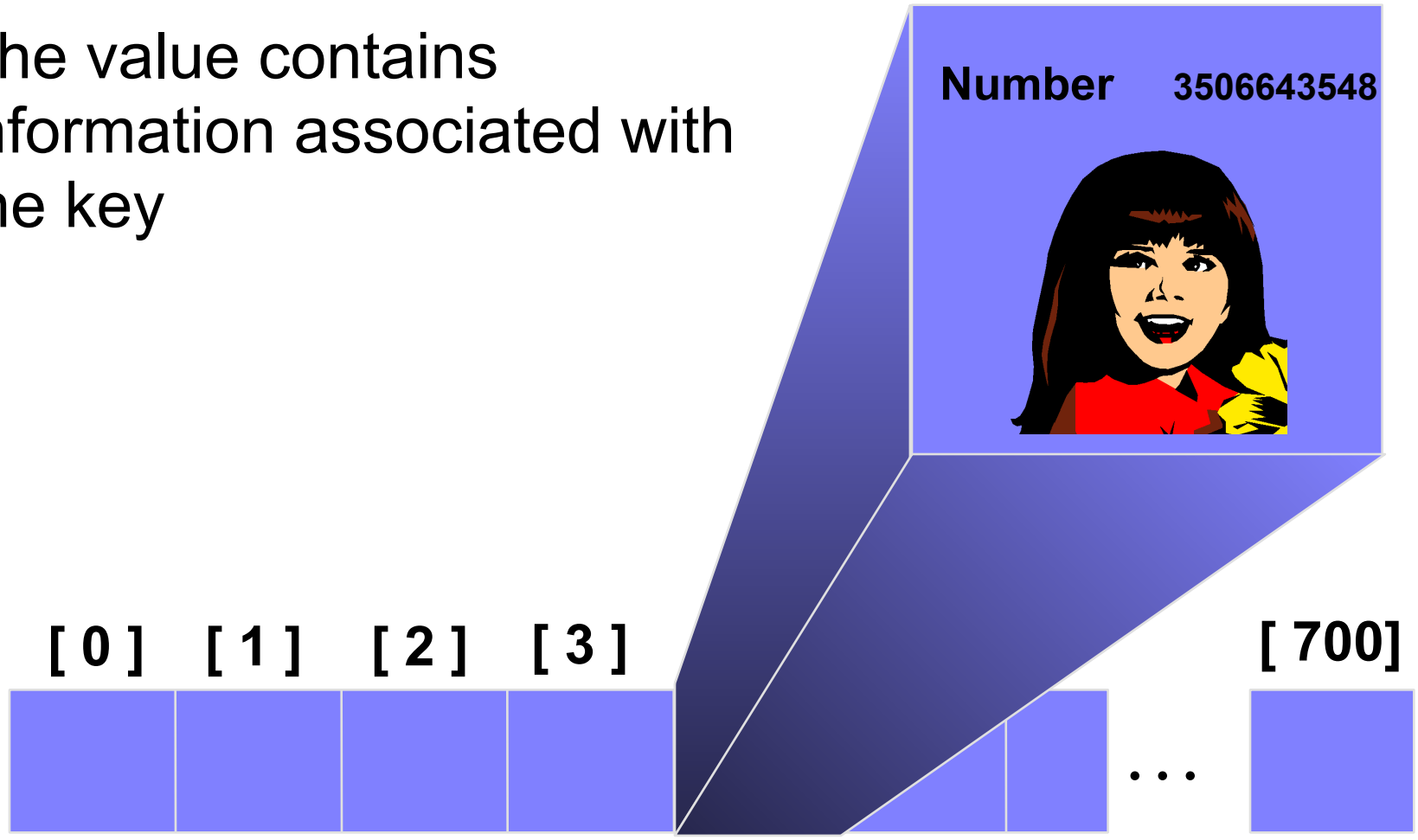
Hash table construction

- Each entry has a special field, called its key.
- In this example, the key is a Social Security Number



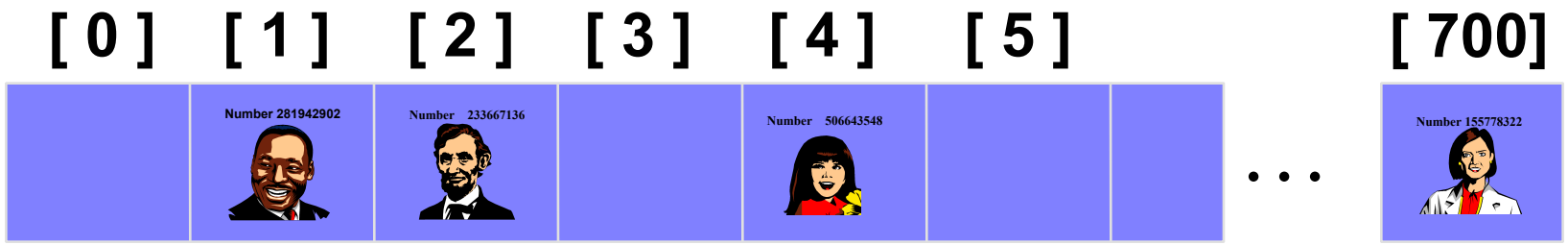
Hash table construction

The value contains information associated with the key



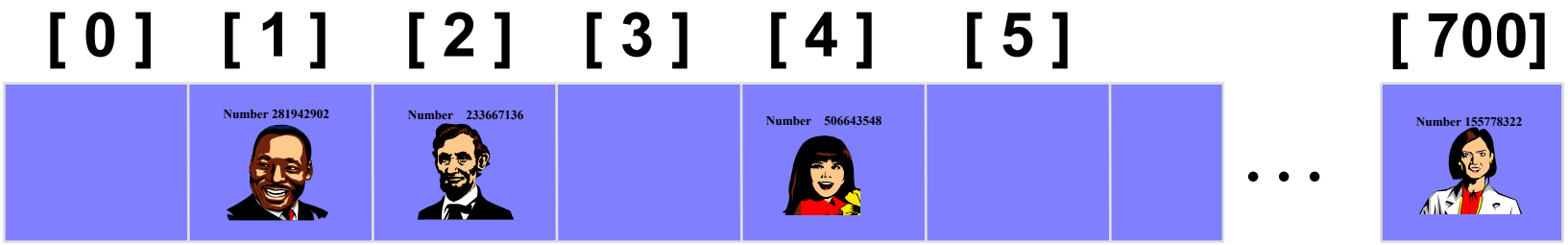
What is a Hash Table ?

When a hash table is in use, some spots contain valid entries, and other spots are "empty".



Inserting a New Entry

- In order to insert a new entry, the **key** must somehow be **converted to** an array **index**.
- The index is called the **hash value** of the key.



Dictionaries

- Operations
  - create
  - destroy
  - insert
  - find
  - delete

- Dictionary: Stores *values* associated with user-specified *keys*

Implementations So Far

	unsorted list	sorted array	Binary search trees (balanced)	
insert				
find				
delete				

Implementations So Far

	unsorted list	sorted array	Binary search trees (balanced)	
insert	find+O(1) = O(n)			
find	O(n)			
delete	find+O(1) = O(n)			

Implementations So Far

	unsorted list	sorted array	Binary search trees (balanced)	
insert	find+O(1) = O(n)	O(n)		
find	O(n)	O(log n)		
delete	find+O(1) = O(n)	O(n)		

Implementations So Far

	unsorted list	sorted array	Binary search trees (balanced)	
insert	find+O(1) = O(n)	O(n)	O(log n)	
find	O(n)	O(log n)	O(log n)	
delete	find+O(1) = O(n)	O(n)	O(log n)	

Implementations So Far

	unsorted list	sorted array	Binary search trees (balanced)	Hash tables
insert	find+O(1) = O(n)	O(n)	O(log n)	O(1)
find	O(n)	O(log n)	O(log n)	O(1)
delete	find+O(1) = O(n)	O(n)	O(log n)	O(1)

Inserting a New Entry

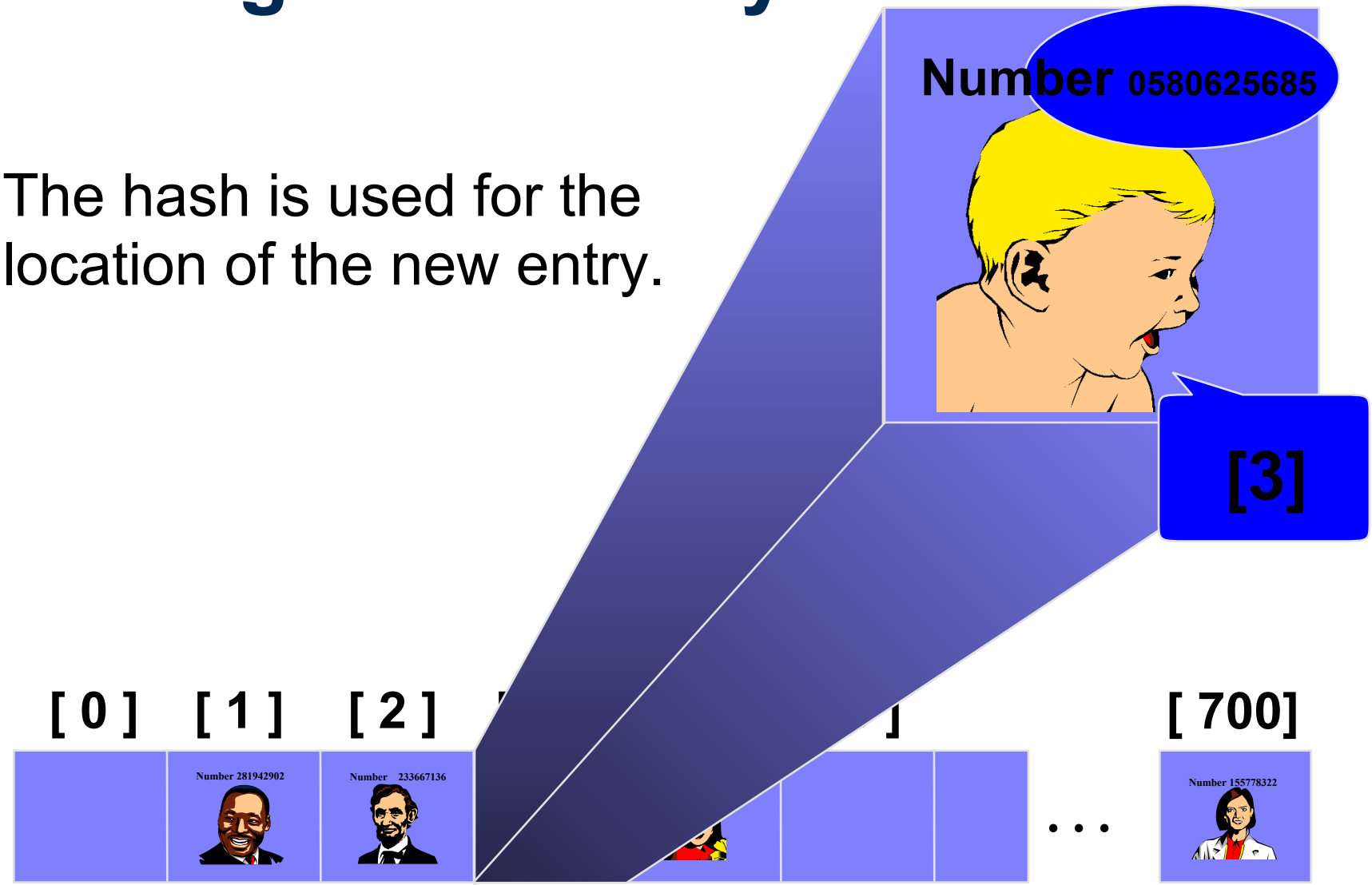
What is `0580625685 % 701`?

Inserting a New Entry

What is `0580625685 % 701`?

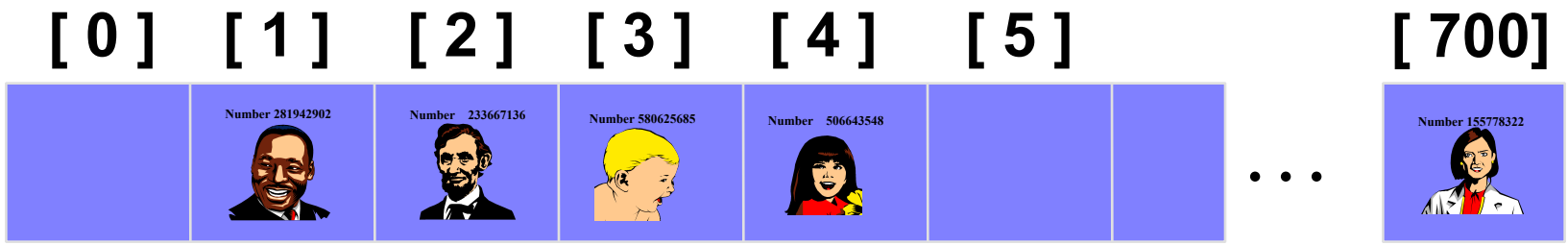
Inserting a New Entry

The hash is used for the location of the new entry.



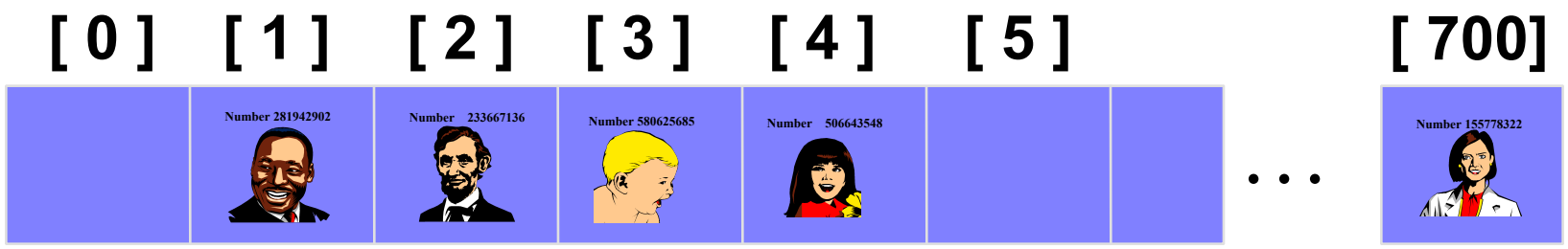
Inserting a New Record

The hash is used for the location of the new entry.



Collisions

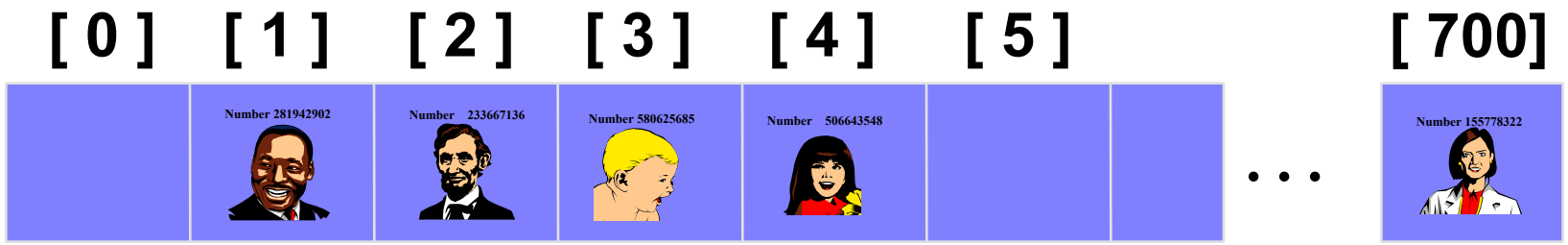
Here is another new entry to insert, with a hash of 2.



Collisions

This is called a **collision**, because there is already another valid entry at [2].

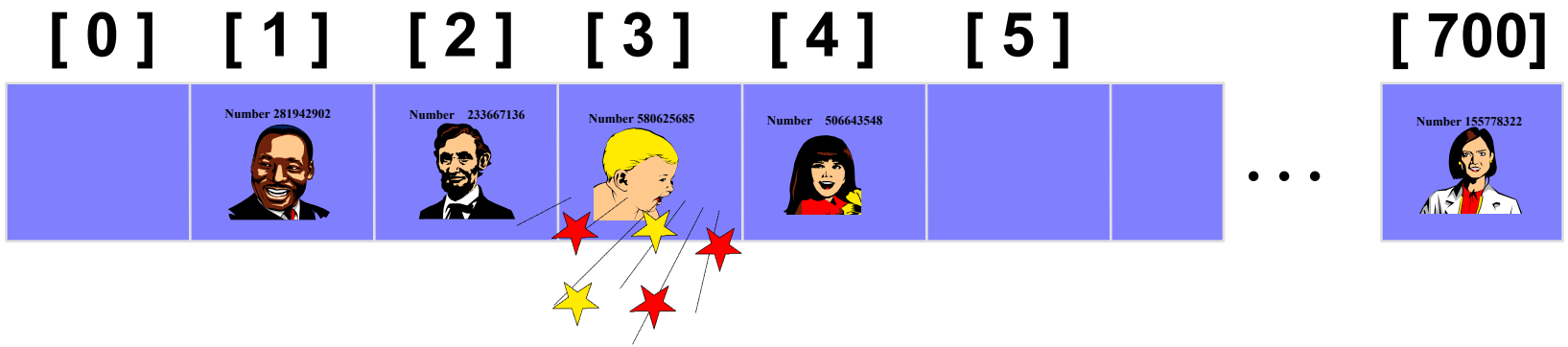
When a collision occurs, move forward until you find an empty spot.



Collisions

This is called a **collision**, because there is already another valid entry at [2].

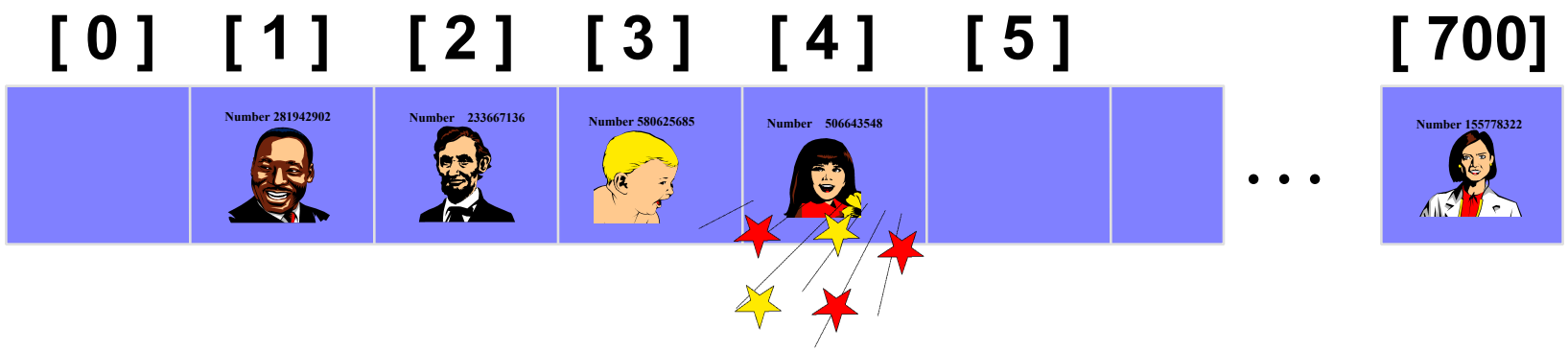
When a collision occurs, move forward until you find an empty spot.



Collisions

This is called a **collision**, because there is already another valid entry at [2].

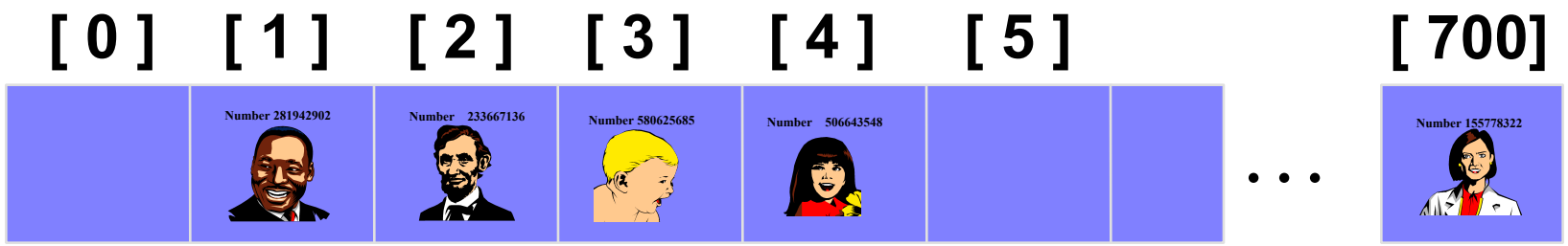
When a collision occurs, move forward until you find an empty spot.



Collisions

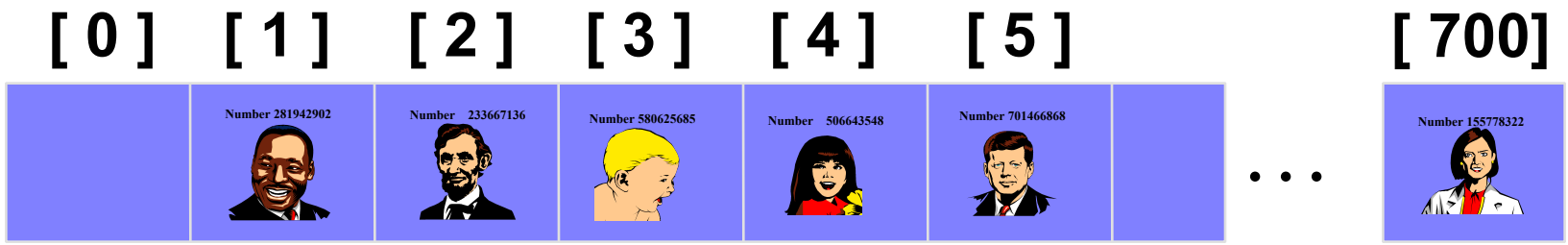
This is called a **collision**, because there is already another valid entry at [2].

When a collision occurs, move forward until you find an empty spot.



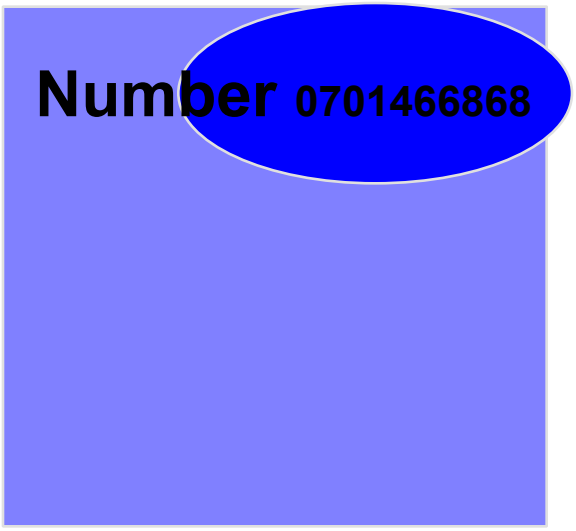
Collisions

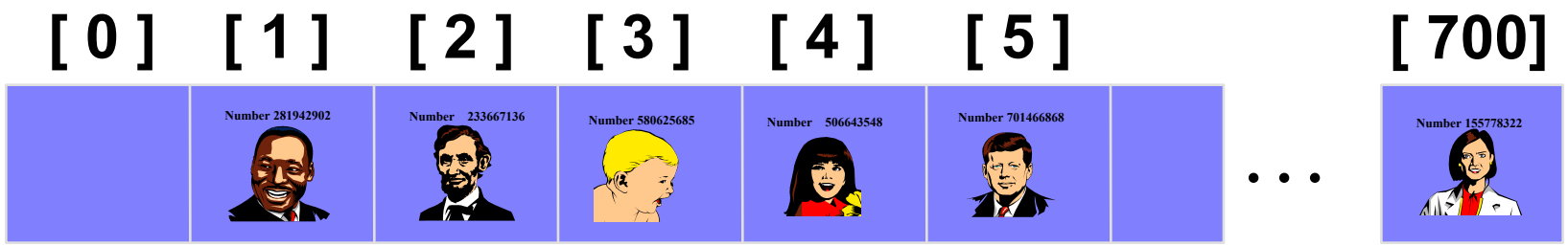
The new data goes in the empty slot



Searching for a Key

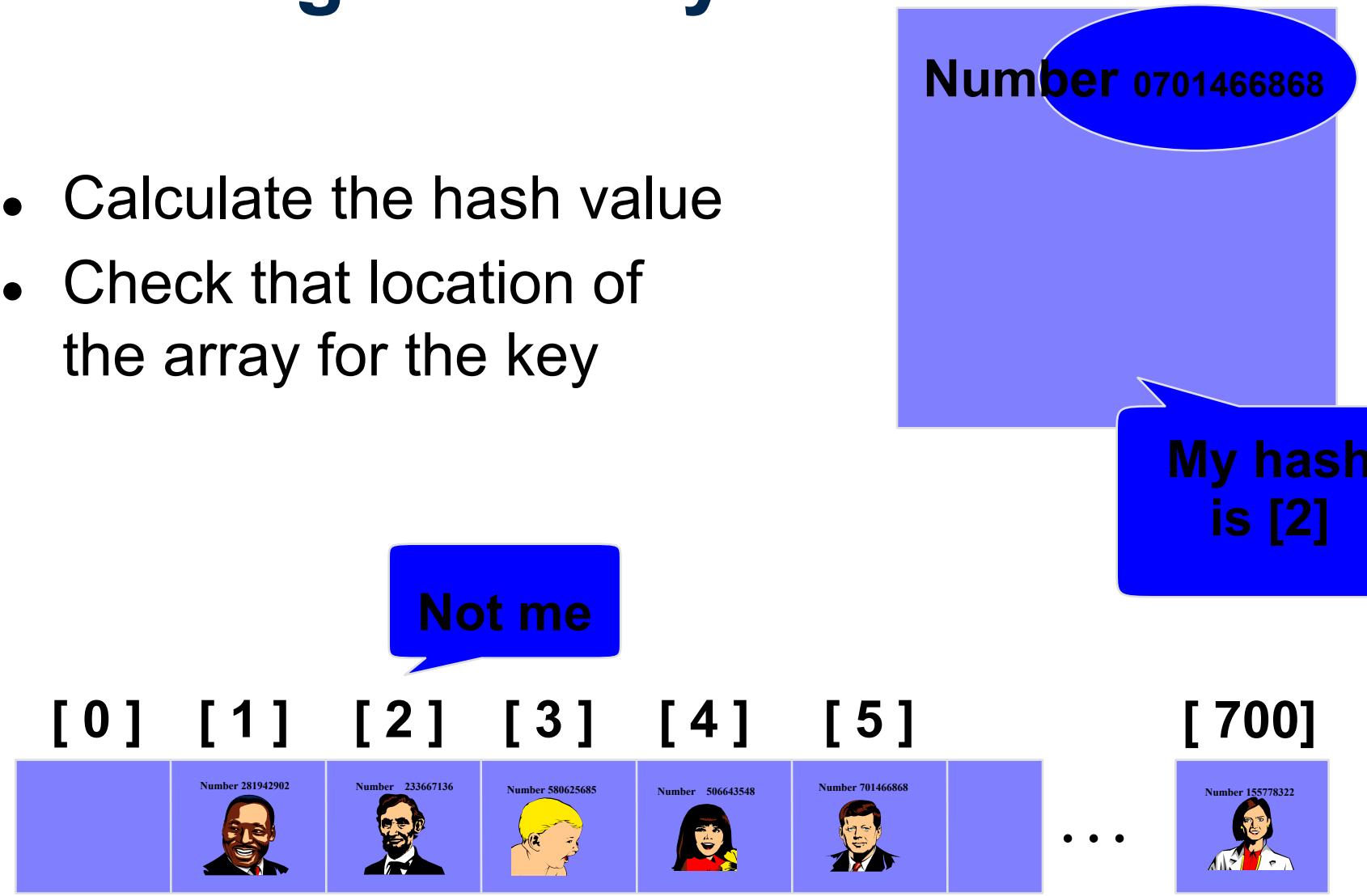
The value that is attached to a key can be found quickly.





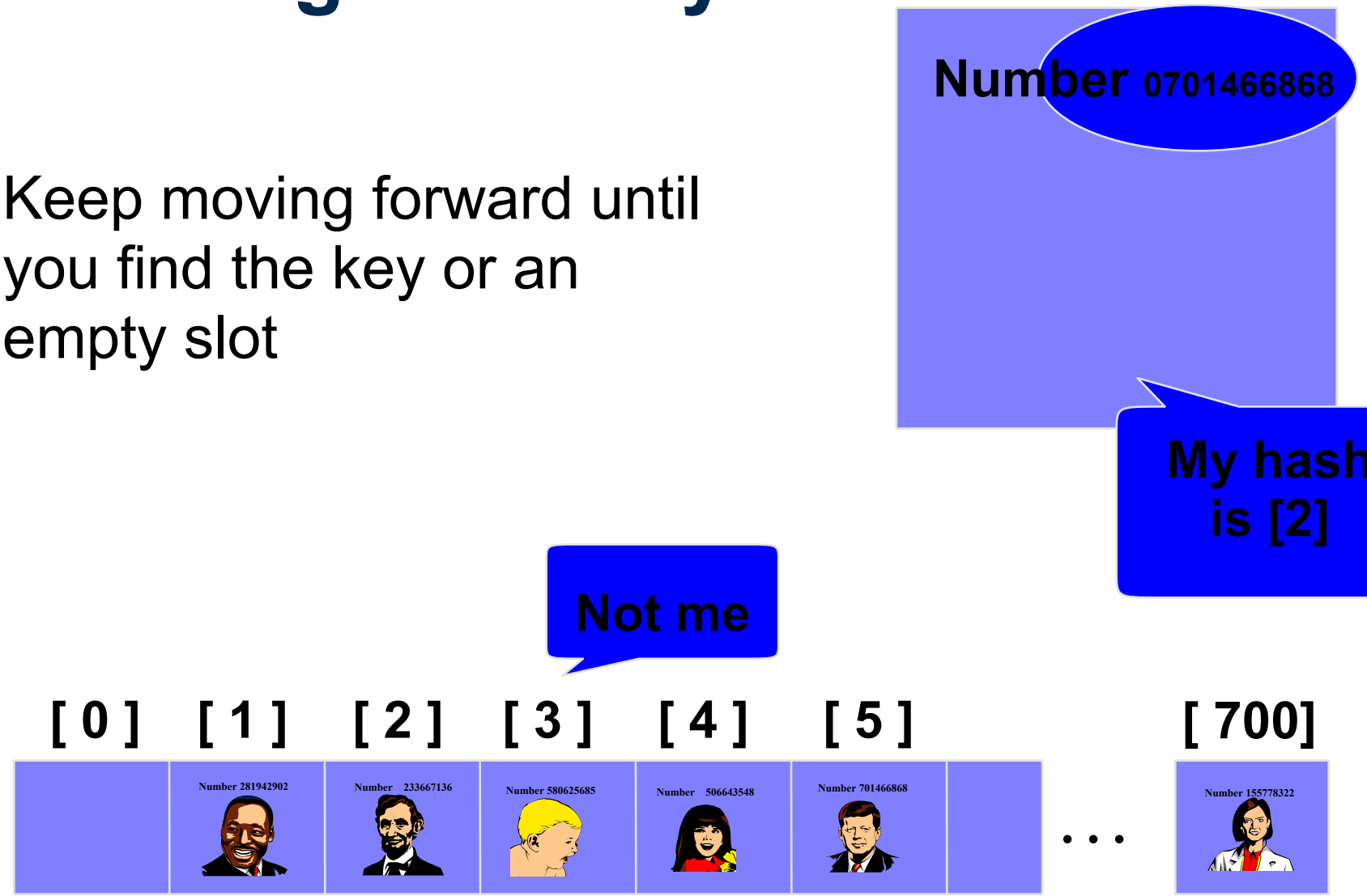
Searching for a Key

- Calculate the hash value
- Check that location of the array for the key



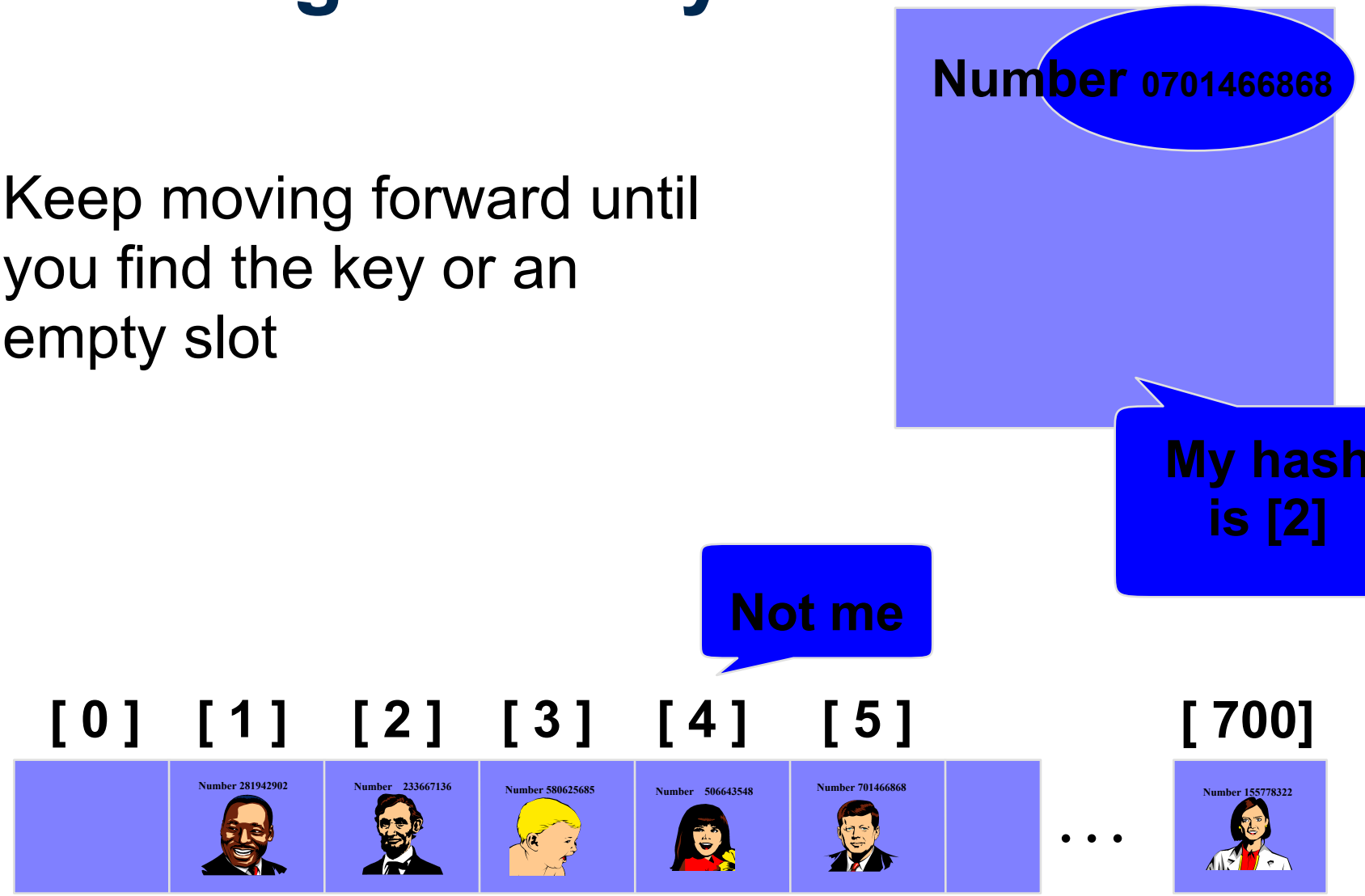
Searching for a Key

Keep moving forward until you find the key or an empty slot



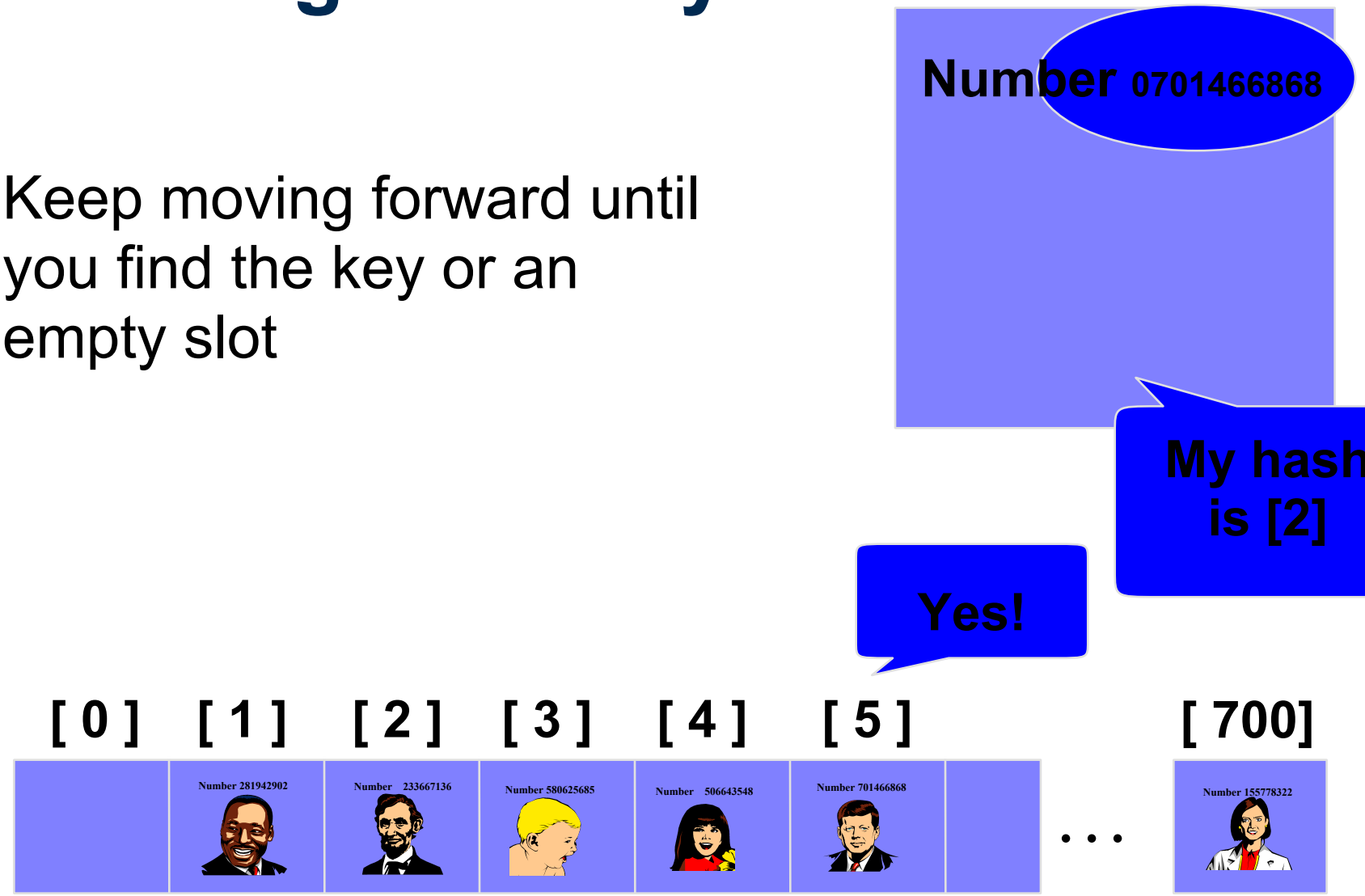
Searching for a Key

Keep moving forward until you find the key or an empty slot



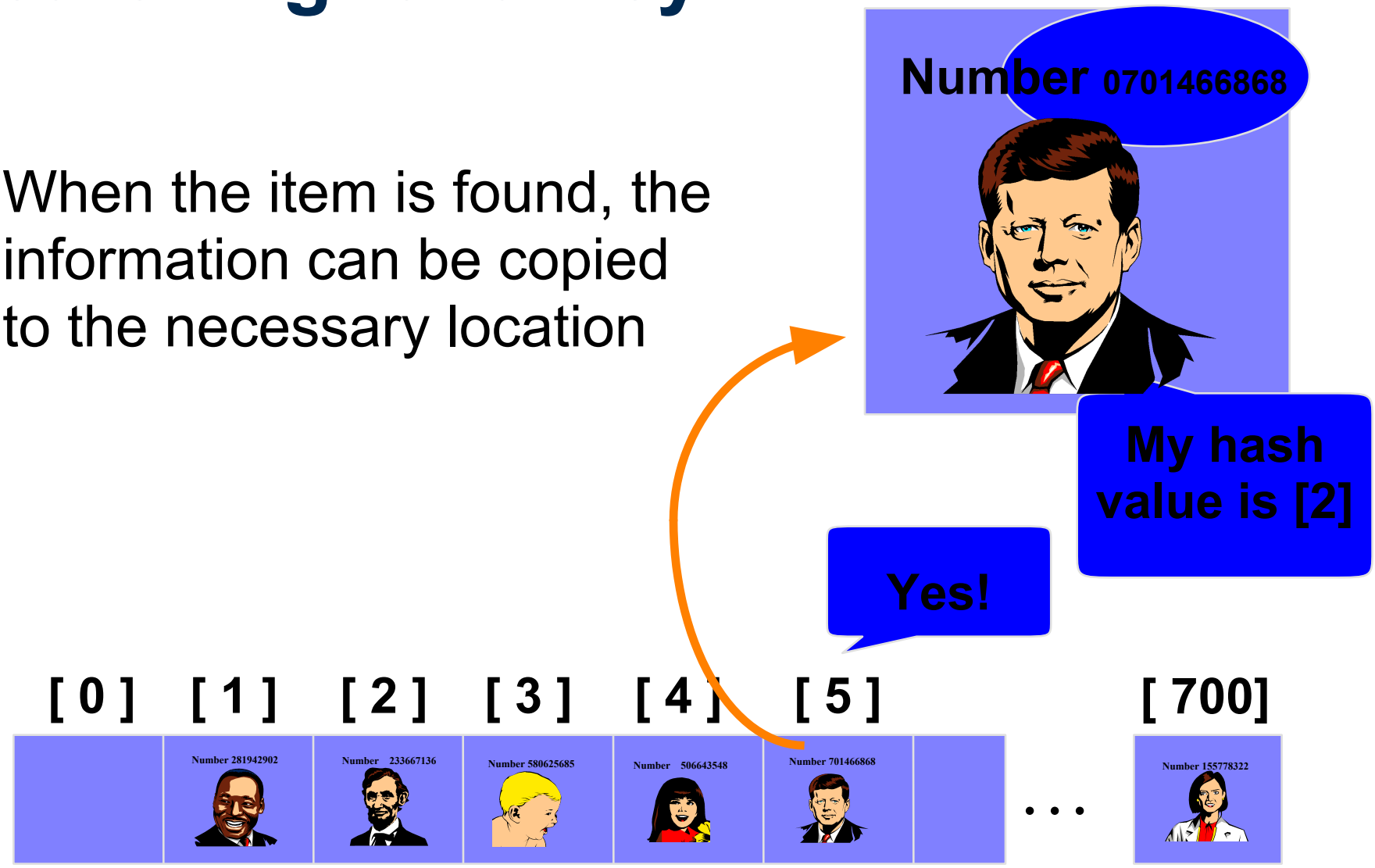
Searching for a Key

Keep moving forward until you find the key or an empty slot



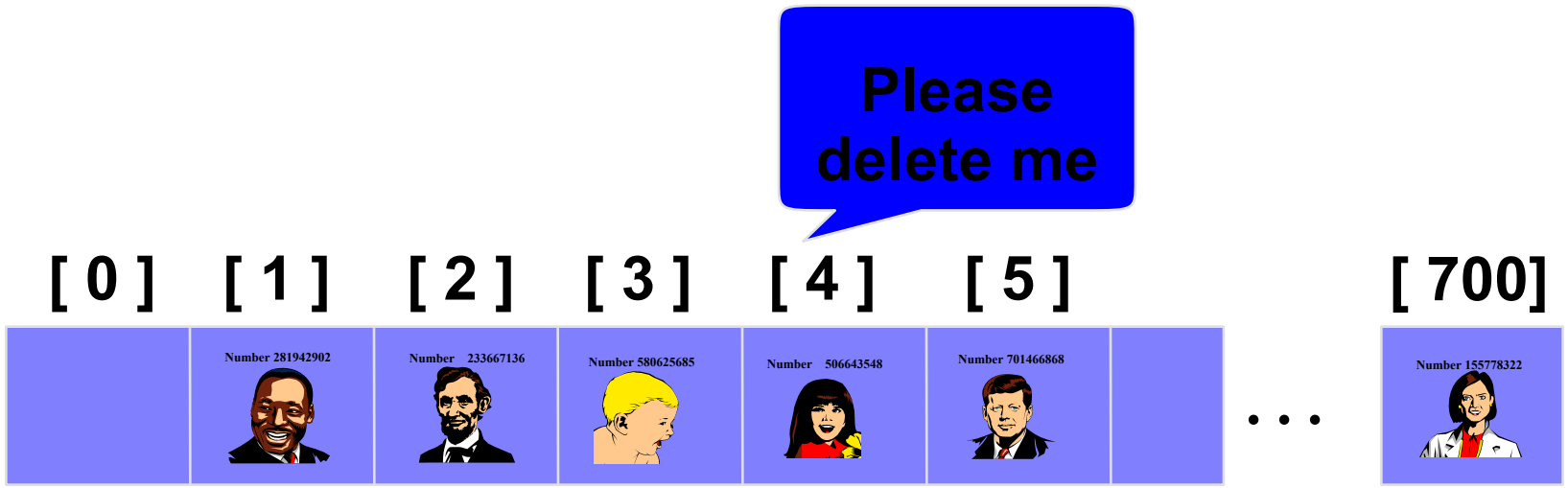
Searching for a Key

When the item is found, the information can be copied to the necessary location



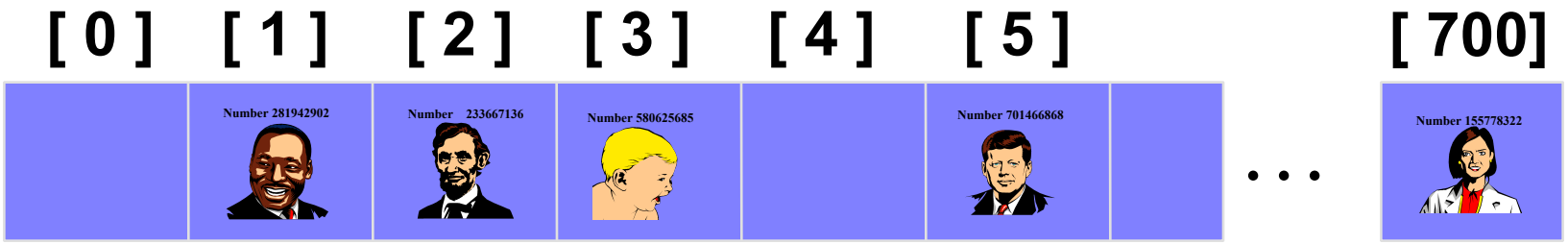
Deleting an Entry

- Entries may also be deleted from a hash table.



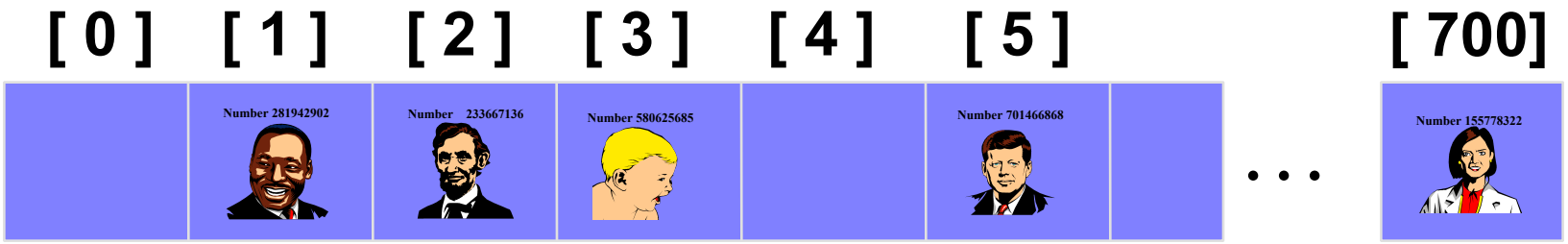
Deleting an Entry

- Entries may also be deleted from a hash table.
- But the location must not be left as an ordinary "empty spot" since that could interfere with searches.



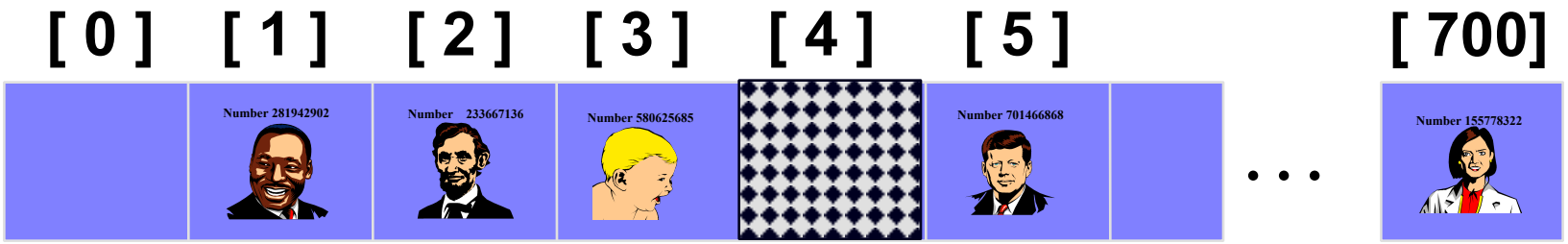
Deleting an Entry

- Entries may also be deleted from a hash table.
- But the location must not be left as an ordinary "empty spot" since that could interfere with searches.



Deleting an Entry

- Entries may also be deleted from a hash table.
- But the location must not be left as an ordinary "empty spot" since that could interfere with searches.
- The location must be marked in some special way so that a search can tell that the spot used to have something in it.



Chances of a collision

- “Birthday paradox”
- What are the odds that two people in 206 have the same birthday
- It turns out to be at more than 50% when we reach 23 people

Chances of a collision

- Suppose our hash table has k entries. The chances of a collision approach 50% when we hash k values

Dictionaries in Python

“items” = “key” : “value”

```
People = {0580625685:'Baby', 0701466868: 'Kennedy',
       3506643548: 'Susan'}

print People[0580625685]
Baby

for i in People:    versus    for i in sort(People):
```

Adding and deleting dictionary entries

```
dict['Key'] = 'New value'; # update existing entry
                           # or add new entry


del dict['Key'];  # remove entry with key 'Key'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary
```

Additional Resources

http://interactivepython.org/runestone/static/pythonds/SortSearch/Hashing.html

http://www.cs.uregina.ca/Links/class-info/210/Hash/